ASG
Software Solutions

ASG-DataManager™ ADABAS Interface Guide

Version 2.5

Publication Number: DMR0200-25-ADA
Publication Date: April 1985

ASG Worldwide Headquarters Naples Florida USA | asg.com | info@asg.com
1333 Third Avenue South, Naples, Florida 34102 USA Tel: 239.435.2200 Fax: 239.263.3692 Toll Free: 800.932.5536 (USA only)

ASG Documentation/Product Enhancement Fax Form

Please FAX comments regarding ASG products and/or documentation to (239) 263-3692.

Company Name	Telephone Number	Site ID	Contact name

Product Name/Publication	Version #	Publication Date
Product:		
Publication:		
Tape VOLSER:		

Enhancement Request:

ASG Support Numbers

ASG provides support throughout the world to resolve questions or problems regarding installation, operation, or use of our products. We provide all levels of support during normal business hours and emergency support during non-business hours. To expedite response time, please follow these procedures.

Please have this information ready:

- Product name, version number, and release number
- List of any fixes currently applied
- Any alphanumeric error codes or messages written precisely or displayed
- A description of the specific steps that immediately preceded the problem
- The severity code (ASG Support uses an escalated severity system to prioritize service to our clients. The severity codes and their meanings are listed below.)
- Verify whether you received an ASG Service Pack for this product. It may include information to help you resolve questions regarding installation of this ASG product. The Service Pack instructions are in a text file on the distribution media included with the Service Pack.

If You Receive a Voice Mail Message:

1. Follow the instructions to report a production-down or critical problem.
2. Leave a detailed message including your name and phone number. A Support representative will be paged and will return your call as soon as possible.
3. Please have the information described above ready for when you are contacted by the Support representative.

Severity Codes and Expected Support Response Times

Severity	Meaning	Expected Support Response Time
1	Production down, critical situation	Within 30 minutes
2	Major component of product disabled	Within 2 hours
3	Problem with the product, but customer has work-around solution	Within 4 hours
4	"How-to" questions and enhancement requests	Within 4 hours

ASG provides software products that run in a number of third-party vendor environments. Support for all non-ASG products is the responsibility of the respective vendor. In the event a vendor discontinues support for a hardware and/or software product, ASG cannot be held responsible for problems arising from the use of that unsupported version.

Business Hours Support

Your Location	Phone	Fax	E-mail
United States and Canada	800.354.3578	239.263.2883	support@asg.com
Australia	61.2.9460.0411	61.2.9460.0280	support.au@asg.com
England	44.1727.736305	44.1727.812018	support.uk@asg.com
France	33.141.028590	33.141.028589	support.fr@asg.com
Germany	49.89.45716.222	49.89.45716.400	support.de@asg.com
Singapore	65.6332.2922	65.6337.7228	support.sg@asg.com
All other countries:	1.239.435.2200		support@asg.com

Non-Business Hours - Emergency Support

Your Location	Phone	Your Location	Phone
United States and Canada	800.354.3578		
Asia	65.6332.2922	**Japan/Telecom**	0041.800.9932.5536
Australia	0011.800.9932.5536	**Netherlands**	00.800.3354.3578
Denmark	00.800.9932.5536	**New Zealand**	00.800.9932.5536
France	00.800.3354.3578	**Singapore**	001.800.3354.3578
Germany	00.800.3354.3578	**South Korea**	001.800.9932.5536
Hong Kong	001.800.9932.5536	**Sweden/Telia**	009.800.9932.5536
Ireland	00.800.9932.5536	**Switzerland**	00.800.9932.5536
Israel/Bezeq	014.800.9932.5536	**Thailand**	001.800.9932.5536
Japan/IDC	0061.800.9932.5536	**United Kingdom**	00.800.9932.5536
		All other countries	1.239.435.2200

ASG Web Site

Visit http://www.asg.com, ASG's World Wide Web site.

Submit all product and documentation suggestions to ASG's product management team at http://www.asg.com/asp/emailproductsuggestions.asp.

If you do not have access to the web, FAX your suggestions to product management at (239) 263-3692. Please include your name, company, work phone, e-mail ID, and the name of the ASG product you are using. For documentation suggestions include the publication number located on the publication's front cover.

Contents

Preface

This *ASG-DataManager ADABAS Interface* publication describes the ADABAS Interface facility, which enables the user to include ADABAS database and file data definitions in the data dictionary, to produce ADABAS LOADER definition cards, and to produce data description statements for ADABAS record buffers and format buffers in COBOL, PL/I, or BAL. Record layouts for ADABAS files and buffers can also be produced.

ASG welcomes your comments, as a preferred or prospective customer, on this publication or on the ASG-DataManager product (herein called DataManager).

About this Publication

The *ASG-DataManager ADABAS Interface* consists of these chapters:

- Chapter 1, "DataManager/ADABAS Interface Facilities," summarizes the interfaces between DataManager and ADABAS.
- Chapter 2, "ADABAS Databases and DataManager," discusses the concept of ADABAS databases and illustrates how ADABAS can be defined to DataManager.
- Chapter 3, "DataManager Data Definition Statements for ADABAS Databases," gives the specifications of the DataManager data definition statements for an ADABAS environment.
- Chapter 4, "ADABAS Source Language Generation from DataManager," describes the interface between ADABAS and the DataManager Source Language Generation facility.
- Chapter 5, "DataManager/ADABAS Correspondence Tables," specifies the direct relationships that exist between ADABAS and DataManager data definitions.

Publication Conventions

ASG's technical publications use these conventions:

Convention	Represents
ALL CAPITALS	Directory, path, file, dataset, member, database, program, command, and parameter names.
Initial Capitals on Each Word	Window, field, field group, check box, button, panel (or screen), option names, and names of keys. A plus sign (+) is inserted for key combinations (e.g., Alt+Tab).
`lowercase italic monospace`	Information that you provide according to your particular situation. For example, you would replace *`filename`* with the actual name of the file.
`Monospace`	Characters you must type exactly as they are shown. Code, JCL, file listings, or command/statement syntax. Also used for denoting brief examples in a paragraph.

Requesting Publication Changes

Customers and other ASG departments can use a Documentation Correction/Enhancement Request Form to request corrections, updates, and enhancements to publications. The form is included in the front matter of each publication. Forms are also available from the Vice President of Technical Publications.

The Vice President of Technical Publications evaluates requests for documentation changes.

1 DataManager/ADABAS Interface Facilities

These interface facilities between DataManager and ADABAS are provided by ASG for the ADABAS user:

- The ability to define ADABAS databases to DataManager, to hold the definitions in the data dictionary, and to process them by the standard DataManager commands
- The ability to generate definition cards that are used by the ADABAS LOADER utility for the initial definition and loading of an ADABAS file
- The ability to generate from the data dictionary, and to insert into the required source library, record buffer layouts and format buffer layouts, which are compiled into user programs and utilised by ADABAS at program execution time.

The ability to define an ADABAS database demands a further member type in DataManager. This member type is ADABAS-DATABASE, which in the member type hierarchy comes between MODULE and FILE. Also required are specific extensions to the FILE data definition statement, to cater for ADABAS files. Additionally, facilities are required to allow the processing view of a database to be defined; these are provided at the SYSTEM, PROGRAM, and MODULE data definition levels. The ADABAS-DATABASE data definition statement and the relevant formats of the FILE and the SYSTEM, PROGRAM and MODULE data definition statements are further discussed in Chapter 2, "ADABAS Databases and DataManager," on page 3 and specified in Chapter 3, "DataManager Data Definition Statements for ADABAS Databases," on page 7.

So that the definitions of ADABAS databases and files may be processed by DataManager in the same way as other members of the data dictionary, the keywords ADABAS-DATABASES and ADABAS-FILES are added to the member-type keywords available for use in these commands:

- BULK
- GLOSSARY
- LIST
- PERFORM
- REPORT
- WHICH.

In addition, the following keywords are available for use within the VIA clause of the WHICH and WHAT commands:

- ACCESSES
- GIVING
- USING
- EDIT-NAME
- COUNTS-AS
- COUPLE
- CIPHER
- DESCRIPTORS
- SUB-DESCRIPTORS
- SUPER-DESCRIPTORS
- FIELD-NAMES
- PHONETIC-NAMES

These keywords are additional to those described in the specifications of the WHICH and WHAT commands. Apart from the keyword ACCESSES, which is available for use with the IDMS Interface and the System 2000/80 Interface, these keywords are specifically for use with the ADABAS Interface. These points regarding their use should be noted:

- The keywords GIVING and USING have the same meaning
- An interrogation that includes VIA GIVING or VIA USING refers to all attributes or subordinate clauses of the GIVING or USING clause, except for the EDIT-NAME and the COUNTS-AS subordinate clauses

The ability to generate ADABAS LOADER Definition Cards, record buffer layouts and format buffer layouts from data dictionary members requires the use of the Source Language Generation facility, which is described in a separate manual. The Source Language Generation manual describes the basic version of the facility, which can output data descriptions in COBOL, PL/I, or BAL. The enhancements to the facility to enable it to generate ADABAS specific outputs are discussed in Chapter 4, "ADABAS Source Language Generation from DataManager," on page 21.

2

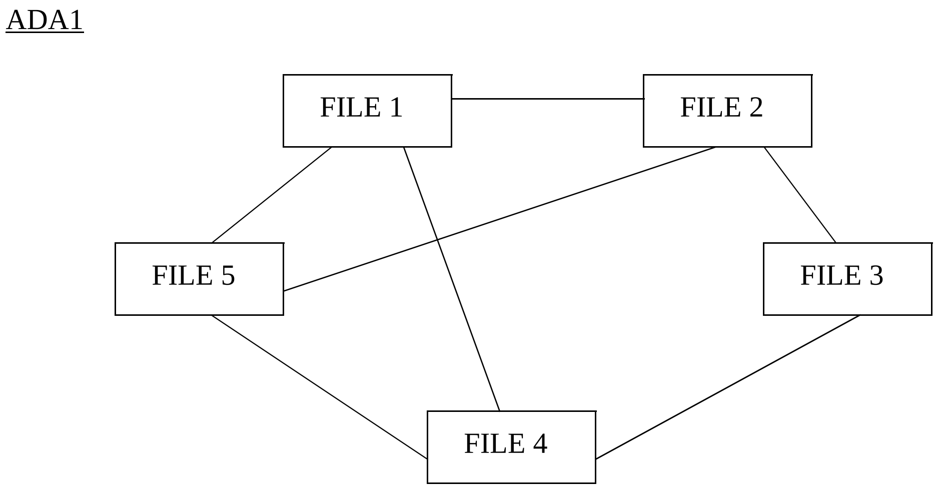
ADABAS Databases and DataManager

Introduction

The ability to give a complete definition of an ADABAS database is available within DataManager.

From a data viewpoint the ADABAS system is a collection of up to 255 files referred to as a database. Within this database various links can be established between the stored data.

Consider, for example, a database as follows:



The database named ADAl consists of the five files, FILE 1, FILE 2, FILE 3, FILE 4, and FILE 5. There are seven established links (defined by ADABAS Descriptors or sometimes File Coupling) that show how the data in those files is logically connected. Thus by accessing data from FILE 1, data from all, some or none of the files FILE 2, FILE 4, and FILE 5 will be accessed. It is, of course, possible that because the files FILE 2 and FILE 3 are linked then accessing data in FILE 1 could also cause access to the data in FILE 3.

For DataManager the above structure would be represented as an ADABAS-DATABASE member containing five FILE members. The linkages between the files would be specified in the ADABAS-DATABASE member. The separate FILE members would contain details of the file contents. The fields which constitute the files would be defined via the normal DataManager GROUP and ITEM data definition statements.

This is a simplified look at the database side of the stored data. Still to be defined is the way in which the data is accessed. Any given program which accesses an ADABAS database is extremely unlikely to access every single field stored in the database. Even if a program accesses one or more specific files in the database it is still unlikely to want each field in those files. Furthermore, a given program may have restrictions placed on it, in that it may only read certain fields, update other fields1 etc. A DataManager SYSTEM, PROGRAM, or MODULE data definition statement can be used to define which databases, files, and fields are processed in an application, together with field sensitivity.

Thus, to define ADABAS databases to DataManager, the ADABAS-DATABASE, FILE, and SYSTEM, PROGRAM, or MODULE member types are used. DataManager GROUP and ITEM members define the fields in the database files.

If certain assumptions are made regarding specific attributes of the database in the above example, then the following would be the method of using DataManager data definition statements to describe the database and one application which accesses the database. The data definitions shown can be inserted into the data dictionary by INSERT or ADD commands, in the same way as any other DataManager data definitions.

```
ADD ADA1;
ADABAS-DATABASE
CONTAINS FILE1 1, FILE2 2, FILE3 3, FILE4 4, FILE5 5
DEVICE DISK 3330
COUPLE FILEl TO FILE2, FILE4, FILE5
           BY PERSONNEL-NUMBER,
FILE2 TO FILE3, FILE5 BY SALARY,
FILE3 TO FILE4 BY AGE,
FILE4 TO FILE5 BY CITY
CIPHER FILEl BY 10, FILE3 BY 12586
;

ADD FILE1;
FILE ADABAS
CONTAINS PERSONNEL-NUMBER, NAME,
   ADDRESS-LINE1, ADDRESS-LINE2, ADDRESS-LINE3,
   CITY, STATE, ZIP-CODE
DESCRIPTORS PERSONNEL-NUMBER, CITY
;

ADD FILE2;
FILE ADABAS
CONTAINS PERSONNEL-NUMBER,
   SOCIAL-SECURITY-NUMBER, TAX-CODE,
   SALARY, (12) MONTHLY-PAY
DESCRIPTORS PERSONNEL-NUMBER, SALARY
;
```

```
FILE3, FILE4, and FILE5 similar to FILEl and FILE2.

ADD PAYE99;
PROGRAM
LANGUAGE 'COBOL'
PROCESSES ADABAS
  ACCESS CALL1 FILE FILEl
    USING      PERSONNEL-NUMBER EDIT-CODE 'El',
               CITY, (3) ' ',
               STATE
  ACCESS CALL2 FILE FILE2
    GIVING     PERSONNEL-NUMBER,
               MONTHLY-PAY OCCURRENCES 1 THRU 6
;
```

Further Considerations

ADABAS Fields

DataManager GROUPs and ITEMs are used to represent the fields contained in the FILEs constituting an ADABAS-DATABASE. A GROUP can contain ITEMs and/or other GROUPS.

ADABAS multiple value fields are defined in the CONTAINS clauses of FILE or GROUP members, as multiple occurrences of an ITEM member; that is, in the form (occurs bound) item-name.

ADABAS periodic group fields are defined in the CONTAINS clauses of FILE or GROUP members, as multiple occurrences of a GROUP member; that is, in the form (occurs bound) group-name.

Names of ADABAS Entities

DataManager provides three methods of naming entities used in ADABAS databases.

ADABAS entities (fields, records, files, databases) are defined to DataManager as ITEMs, GROUPs, FILEs, or ADABAS-DATABASEs and their data definitions are inserted into a data dictionary as members of that dictionary. When each member is inserted, it is given a name; this name is its member name. It is unlikely that the member name will be the same as the name given to the entity within ADABAS. The reason for this is that ADABAS names are cryptic and restricted in length and format. The member name, however, will normally be meaningful and has a larger length limit (32 characters) and a fairly unrestricted format.

The second method of naming an ADABAS entity is provided by the ALIAS clause, which is available in all DataManager data definition statements. This clause can be used to define a specific ADABAS name as an alias of the member. The Source Language Generation facility can apply the ADABAS aliases (instead of the member names) to the generated entities if the user so specifies.

Ideally an ADABAS field/record name ought to be consistent throughout an organisation. Thus, for example, whenever "A2" is referred to it is unambiguous as it always indicates the ADABAS alias for a specific ITEM. However, this is not always possible, so a third naming technique is provided when an ADABAS file is defined to DataManager. This allows a specific ADABAS name to be associated with a specific ITEM or GROUP for the file being defined. See the specification of the KNOWN-AS local-name clause, in "The File Data Definition Statement for ADABAS Files" on page 10.

3 DataManager Data Definition Statements for ADABAS Databases

The ADABAS-DATABASE Data Definition Statement

Format

```
ADABAS-DATABASE
[CONTAINS filename-entry [,filename-entry]. . .]
[DEVICE ⎧ xxxx [yyyy] ⎫]
        ⎩ yyyy        ⎭
[COUPLE filename ⎧ TO   ⎫ filename [,filename]. . .BY item-name
                 ⎩ WITH ⎭
[,filename ⎧ TO   ⎫ filename [,filename]. . .BY item-name]. . .]
           ⎩ WITH ⎭
[CIPHER filename [,filename]. . .⎧ BY   ⎫⎧ code        ⎫
                                 ⎩ WITH ⎭⎩ item-name-a ⎭
[,filename [,filename]. . .⎧ BY   ⎫⎧ code        ⎫ ]. . .]
                           ⎩ WITH ⎭⎩ item-name-a ⎭
[common clauses]
⎧ ; ⎫
⎩ . ⎭
```

where:

filename-entry is defined as:

filename [*adabas-code*]

where:

filename is the name of a file within the ADABAS database

adabas-code is an integer in the range 1 to 255, being the ABABAS file number of the file within the database being defined. The *adabas-code* must either be present in all *filename-entries*, or be present in none.

xxxx is a 4-character generic device type

yyyy is a 4-digit specific device type *item-name* is the name of an item that is defined as a descriptor in all of the files being coupled together

code is a 1- to 8-digit integer used by ADABAS to generate a code table which is used to cipher the data stored in the ADABAS file

item-name-a is an item capable of holding a code (where code is as defined above)

common clauses are any of the following clauses, in any order:

ACCESS-AUTHORITY
ADMINISTRATIVE-DATA
ALIAS
CATALOGUE
COMMENT
DESCRIPTION
EFFECTIVE-DATE
FREQUENCY
NOTE
OBSOLETE-DATE
QUERY
SECURITY-CLASSIFICATION
SEE

Remarks

1. Filenames must conform to the rules for member names.

2. The CONTAINS keyword is followed by a list of from one to 255 filename-entries. If two or more filename-entries are listed, each except the last in the list must be followed by a comma. Filename-entries may be additionally separated from each other by spaces. Only one CONTAINS clause can appear.

3. If *adabas-code* is not specified in the filename-entries, then on encoding DataManager generates *adabas-code* numbers sequentially from 1 for the files named in the CONTAINS clause. If *adabas-code* is specified in the first *filename-entry* but is omitted from a subsequent *filename-entry*, the error message:

   ```
   ADABAS-CODE ENTRY MISSING OR INCOMPLETE
   ```

   is output, and encoding fails. If *adabas-code* is not specified for the first *filename-entry* but is specified for a subsequent one, the warning message:

   ```
   adabas-code SUPERFLUOUS FILE CODE - IGNORED
   ```

   is output, and encoding proceeds with the DataManager generated *adabas-code* numbers applied.

4. If a four character generic device type is stated in the DEVICE clause, it must be one of these:

   DASD (for Direct Access Storage Device)
   DISK
   DRUM

5. If a four digit specific device type is stated in the DEVICE clause, it must be one of these:

   2305
   2314
   3330
   3340
   3350
   4580

   These are the devices on which ADABAS databases may be held. Only one DEVICE clause can appear. If the DEVICE clause is omitted, a default of DISK 2314 is taken.

6. COUPLE clauses specify the files that are coupled together on the database, and the descriptors used to achieve the coupling. Any number of COUPLE clauses may appear. A filename is coupled to one or more other filenames by an *`item-name`* (*`item-name`* must be defined as a DESCRIPTOR in all of the files concerned). A filename may be coupled to a maximum of 180 other filenames. Any two filenames may only be coupled by a single item-name.

   The keyword TO or WITH is followed by a list of from one to 180 filenames; each filename except the last in the list must be followed by a comma and may optionally additionally be followed by spaces. The last filename in the list is followed by the keyword BY and an *`item-name`*. If the same file is coupled to another list of filenames, *`item-name`* is followed by a comma and optionally additionally by spaces. Coupling for a new filename is specified in a new COUPLE clause. Coupling is defined one-way only (for example, if FILE1 is coupled to FILE2 it is not necessary to specify that FILE2 is coupled to FILE1).

7. CIPHER clauses specify the cipher code used by ADABAS storing data in the specified filename. Any number of CIPHER clauses may appear, but only one CIPHER clause can appear for each filename.

8. Because of the sensitive nature of the information in the CIPHER clauses, it is recommended that the Controller issue a PROTECT command to prevent other users of the data dictionary from accessing the definition of the database; alternatively, an *`item-name`* may be specified in the CIPHER clause, and the Controller may PROTECT the item against access by other users. (The Controller would define a HELD-AS form for the item with a CONTENTS clause giving the actual cipher value.)

9. *`Common clauses`*, listed under Format above, can be present in any type of data definition statement. Not more than one of each of these clauses can be declared. If a common clause has a subordinate clause or keyword, the subordinate clause identifier or subordinate keyword must not be truncated to an extent where it becomes ambiguous with any other clause identifier or other keyword available in the data definition syntax for this member type.

10. A record containing the database’s data definition statement can be inserted into the data dictionary’s source data set by a suitable command and an encoded record can subsequently be generated and inserted into the data entries data set. If, when the encoded record is generated, any file or item whose name appears in the database’s data definition statement has no data entries record, DataManager creates a dummy data entries record for that member. The dummy record is created as a dummy file record except where the name appears in a BY subordinate clause of a COUPLE clause or in a BY or WITH subordinate clause of a CIPHER clause, in which cases a dummy item record is created.

Example

See "Introduction" on page 3.

The File Data Definition Statement for ADABAS Files

Format

```
FILE ADABAS
[CONTAINS content  [,content].. .]
[DESCRIPTORS member-name-d [UNIQUE] [,member-name-d [UNIQUE]]. .
[SUB-DESCRIPTORS adabas-name IS member-name-s BYTES a  [TO b]
  [,adabas-name IS member-name-s BYTES a  [TO b]]. . .]
[SUPER-DESCRIPTORS adabas-name IS member-name-p [BYTES a  [TO b]]
  [WITH member-name-p  [BYTES a  [TO b]]]. . .]. . .]
   [,adabas-name IS member-name-p [BYTES a  [TO b]]
     [WITH member-name-p [BYTES a  [TO b]]. . .
[FIELD-NAMES adabas-name IS { group-name }
                            { item-name  }
            [,adabas-name IS { group-name } ]. . .]. . .
                             { item-name  }
[PHONETIC-NAMES adabas-name IS item-name
                [,adabas-name IS item-name]. . .]
[common clauses]
{ ; }
{ . }
```

where:

`content` declares a group, an item, or an array in the format:

```
{ group-name                                                         }
{ item-name [version]                                                }
{ ({ integer                  }) { group-name           }            }
{  { item-name-a [version]    }  { item-name [version]  }            }
[KNOWN-AS local-name] [INDEXED-BY index-name]
```

where:

`group-name` is the name of a group

`item-name` is the name of an item

`version` is an unsigned integer in the range 1 to 15, being a number specifying which version of the relevant item is relevant to this file. The version is within the HELD-AS form, or within a defaulted form as stated in Remark 1. If version is omitted or if the stated version does not exist the lowest numbered existing version is assumed to be relevant.

`integer` is an unsigned integer of from one to eighteen digits, being the number of times group-name or item-name occurs in the array

`item-name-a` is the name of an item. This form of array declaration declares that when the file is processed by an application program or module, the number of times `group-name` or `item-name` occurs in the array is contained in the item `item-name-a`.

`local-name` is a name, conforming to the rules for member names that can be used instead of the name or alias of the contained member, when ADABAS source statements are generated from this data definition by the DataManager Source Language Generation facility. The `local-name` is not separately recorded in the data dictionary (that is, no dummy data entries record and no index record is created for `local-name` when the data definition in which it appears is encoded) so `local-name` cannot be interrogated and can be the same as another name, an alias or a catalog classification in the data dictionary. The `local-name` is the name by which the contained member is known only within the file defined by this data definition.

`index-name` is a name, conforming to the rules for member names, that is to be used as the index name when COBOL data descriptions are generated by the DataManager Source Language Generation facility. The `index-name` is not separately recorded in the data dictionary (that is, no dummy data entries record and no index record is created for `index-name` when the data definition in which it appears is encoded) so `index-name` cannot be interrogated and can be the same as another name, an alias, or a catalog classification in the data dictionary.

`member-name-d` is the name of an item or group which is to be recognised as a DESCRIPTOR in the file being defined. The specified member must be directly or indirectly contained in the file.

`adabas-name` is the two character name conforming to the ADABAS rules for ADABAS field names.

`member-name-s` is the name of an item or group, part of the contents of which is to be recognised as a sub-descriptor in the file being defined. The specified member must be directly or indirectly contained in the file.

`a` is an integer specifying the byte number of the item at which the sub-descriptor field or super-descriptor field is to start. Bytes are counted from the left or right of the item depending on whether the item is alphanumeric (left) or numeric (right). The first byte number in this count is 1.

b is an integer specifying the last byte number of the item to which the sub-descriptor field or super-descriptor field extends. The value of *b* must not exceed the length in bytes of the item.

member-name-p is the name of an item or group, all or part of the contents of which forms a super-descriptor field in the file being defined. The specified member must be directly or indirectly contained in the file.

group-name is the name of a group

item-name is the name of an item

common clauses are any of the following clauses, in any order:

- ACCESS-AUTHORITY
- ADMINISTRATIVE-DATA
- ALIAS
- CATALOGUE
- COMMENT
- DESCRIPTION
- EFFECTIVE-DATE
- FREQUENCY
- NOTE
- OBSOLETE-DATE
- QUERY
- SECURITY-CLASSIFICATION
- SEE

Remarks

1. Any direct or indirect reference from the CONTAINS clause to an item is assumed to be the HELD-AS form of that item. If the item has no HELD-AS form, default assumptions are made as to the relevant form of the item, in the order DEFAULTED-AS, ENTERED-AS, REPORTED-AS. The form first encountered in this order is taken as the defaulted form, and version is applied within that form as stated under "Format."

2. If the Source Language Generation facility is used to generate LOADER Definition cards or buffer data description statements in respect of this file, and any group or item member that is directly or indirectly referenced from the contains clause has no entry in a FIELD-NAMES clause, then DataManager generates a FIELD-NAMES clause in respect of that member. See the specification of the PRODUCE command in "Specification of the PRODUCE Command for ADABAS Source Language Generation" on page 22.

3. If the same item or group appears more than once in the same file, it is possible to specify in the FIELD-NAMES clause the ADABAS names that are to be used for the second and subsequent occurrences of the item or group when LOADER Definition cards are produced. For example, if these clauses were processed when producing LOADER Definition cards:

   ```
   FIELD-NAMES  Al IS ITEM-A,
                A2 IS ITEM-A,
                A3 IS ITEM-A
   ```

   the first occurrence of ITEM-A would be generated with the ADABAS name Al, the second would be generated with the ADABAS name A2, and so on.

   Note:

   These restrictions apply when the same item or group appears more than once in the same file:

   - It is not possible to define a particular occurrence of the item or group as a descriptor, while other occurrences of the same item or group are not descriptors
   - It is not possible to select the second or a subsequent occurrence of the item or group for use in a record or format buffer.

4. If the Source Language Generation facility is used to generate LOADER Definition cards or Format Buffer data description statements in respect of this file, any group specified in a DESCRIPTOR, SUB-DESCRIPTOR, or SUPER-DESCRIPTOR clause is treated as an item. (If a Record Buffer data description statements are generated in respect of this file these groups are treated as groups.)

5. If the Source Language Generation facility is used to generate LOADER Definition cards in respect of this file, any member whose name appears in the DESCRIPTORS clause and is immediately followed by UNIQUE is treated as a unique descriptor.

6. Common clauses, listed under Format above, can be present in any type of data definition statement; they are therefore defined separately. Not more than one of each of these clauses can be declared. If a common clause has a subordinate clause or keyword, the subordinate clause identifier or subordinate keyword must not be truncated to an extent where it becomes ambiguous with any other clause identifier or other keyword available in the data definition syntax for this member type.

7. A record containing the file’s data definition statement can be inserted into the data dictionary’s source data set by a suitable command and an encoded record can subsequently be generated and inserted into the data entries data set. If, when the encoded record is generated, any item or group whose name appears in the file’s data definition statement has no data entries record, DataManager creates a dummy data entries record for that member. The dummy record is created as a dummy item record.

Example

See "Introduction" on page 3.

System, Program and Module Data Definition Statements for an ADABAS Environment

Introduction

The data definition statements for DataManager SYSTEM, PROGRAM, and MODULE members acting on conventional files are described in another publication. For the ADABAS Interface, a further clause, the PROCESSES ADABAS clause, is included in the format of these data definition statements. This section describes that clause.

Specification of the PROCESSES Clause

Format

```
PROCESSES ADABAS
ACCESS access-name
     ⎧ FILE filename ⎧  GIVING ⎫ details [,details]. . . ⎫
     ⎨               ⎩  USING  ⎭                         ⎬
     ⎩ AS access-name-2  IN process-name                 ⎭
[ACCESS access-name
     ⎧ FILE filename ⎧  GIVING ⎫ details [,details]. . .  ⎫]. . .
     ⎨               ⎩  USING  ⎭                          ⎬
     ⎩ AS access-name-2  IN process-name                  ⎭
```

where:

access-name is a name that is valid in the programming language of the SYSTEM, PROGRAM or MODULE member in whose data definition this PROCESSES clause appears. Each access-name must be unique within the member.

filename is the name of a member that is a FILE ADABAS member

details is as shown in Figure 1 on page 19

where:

item-name and *item-name-b* are the names of ITEM members that are directly or indirectly contained by filename

version is an unsigned integer in the range 1 to 15, being a number specifying which version of the specified or defaulted form of the relevant item is relevant to this GIVING or USING clause. If *version* is omitted, a default is taken as stated in Remark 8.

item-name-a is the name of an ITEM member whose HELD-AS form has a CONTENTS clause that specifies a single literal that is an ADABAS Edit Mask *n* is an unsigned integer in the range 1 to 15. where E*n* specifies an ADABAS Edit Mask

item-name-c is the name of an ITEM member whose HELD-AS form's form-description specifies the length and format of the count field

i and *j* are unsigned integers in the range 1 to 191

group-name-a is the name of a GROUP member with multiple occurrences that is directly contained by filename

g and *h* are unsigned integers in the range 1 to 99

group-name is the name of a GROUP member that is directly or indirectly contained by filename

nn is an unsigned integer in the range 1 to 255, being the number of occurrences of the following literal. If (*nn*) is omitted, a default of (1) is assumed.

literal is a character string of not more than 255 printable and/or non-printable characters

local-name is a name, valid in the programming language of the SYSTEM, PROGRAM or MODULE member in whose data definition this PROCESSES clause appears, that can be used instead of the member name or alias or filler name (or equivalent) to identify *item-name* or *group-name* or the literal when ADABAS source language statements are generated from this data definition by the DataManager Source Language Generation facility. The *local-name* is not separately recorded in the data dictionary (that is, no dummy data entries record and no index record is created for *local-name* when the data definition in which it appears is encoded) so *local-name* cannot be interrogated and can be the same as another name, an alias or a catalog classification in the data dictionary. The *local-name* is the name by which *item-name*, *group-name* or the literal is known only within this processing view. It must be unique within the processing view.

access-name-2 is an access name that appears in the PROCESSES clause of the member specified by *process-name*

process-name is the name of a SYSTEM, PROGRAM, or MODULE member

Remarks

1. The PROCESSES clause must be immediately followed by the keyword ADABAS, to define the context in which the clause applies.

2. The PROCESSES clause can contain any number of ACCESS clauses, each of which includes either a subordinate FILE clause or a subordinate AS clause.

3. Each ACCESS clause defines the processing view of an ADABAS file that is processed by the system, program or module in whose data definition the clause appears. The processing view is defined either directly by a FILE subordinate clause, or indirectly by an AS subordinate clause.

4. No validation is performed on access-name on encoding; but the PRODUCE command of the Source Language Generation facility checks that *`access-name`* is a valid name in the source language being generated.

5. The GIVING and USING keywords are synonymous to DataManager.

6. The details defined in the GIVING or USING subordinate clause are used by the Source Language Generation facility in the generation of data description statements for ADABAS record buffers and format buffers. The rules and restrictions applying to ADABAS format definitions must therefore be applied when specifying a GIVING or USING clause. When format buffers are produced, statements are generated only for item-names and group-names directly specified in the GIVING or USING clause. When record buffers are produced, statements are generated both for *`item-names`* and *`group-names`* specified in the GIVING or USING clause, and for the groups and items directly or indirectly contained by the *`group-names`*.

7. If no form is specified for item-name (where permitted in the Format specification) in the GIVING or USING clause, a default of HELD-AS is assumed. If the item has no HELD-AS form, default assumptions are made as to the relevant form of the item, in the order DEFAULTED-AS, ENTERED- AS, REPORTED-AS. The form first encountered in this order is taken as the defaulted form, and version is applied within that form. That is, unless overridden by a form declaration in the GIVING or USING clause, the form of *`item-name`* contained by filename is assumed to apply.

8. If no version is stated for *`item-name`* (where permitted in the Format specification) in the GIVING or USING clause, the version of *`item-name`* contained by filename is assumed to apply. If no version is stated for *`item-name-a`* or *`item-name-c`*, the lowest numbered version is assumed to apply.

9. If the Source Language Generation facility is used to generate data description statements for ADABAS format buffers, then:

   - If EDIT-NAME is specified, the literal from the CONTENTS IS clause of the HELD-AS form of *item-name-a* is used as the Edit Mask for the format definition(s) generated for *item-name* or *group-name*. If *item-name-a* has no HELD-AS form, default assumptions are made as to the relevant form, in the order DEFAULTED-AS, ENTERED-AS, REPORTED-AS. The form first encountered in this order is taken as the defaulted form, and version is applied within that form.

   - If EDIT-CODE is specified, the Edit Mask defined by E*n* is used for the format definition(s) generated for *item-name* or *group-name*.

   - If COUNTS is specified, format definitions are generated as count definitions. If *item-name-c* has no HELD-AS form, default assumptions are made as to the relevant form, in the order DEFAULTED-AS, ENTERED- AS, REPORTED-AS. The form first encountered in this order is taken as the defaulted form, and version is applied within that form. If AS *item-name-c* is omitted, a default count field specification of one byte, binary, is assumed.

   - If *item-name* OCCURRENCES is specified, *item-name* is a multiple value field, and the OCCURRENCES clause specifies its occurrence indices

   - If *group-name* OCCURRENCES is specified, *group-name* is a periodic group, and the OCCURRENCES clause specifies its occurrence indices

   - If an IN clause is specified it denotes that *item-name* or *group-name* is contained in a periodic group, *group-name-a*, whose occurrence indices are specified in the IN clause's subordinate OCCURRENCES clause.

   - If *item-name* OCCURRENCES is specified with IN *group-name-a* OCCURRENCES, this indicates that *item-name* is a multiple value field contained in the periodic group *group-name-a*. The OCCURRENCES clauses specify the occurrence indices of *item-name* and *group-name-a*.

   - If OCCURRENCES i [,i]... or OCCURRENCES g [,g]... is specified, a format definition is generated for each specified occurrence index

   - If OCCURRENCES i THROUGH j or OCCURRENCES g THROUGH h is specified, a range specification is generated. The keywords THROUGH and THRU are synonymous.

   - If *item-name* THROUGH *item-name-b* is specified, a range specification is generated. The keywords THROUGH and THRU are synonymous. The range of items involved is the range of items from *item-name* to *item- name-b* as they occur in filename's records; that is, as they are directly specified in or indirectly specified via the CONTAINS clause of filename. Hence, *item-name-b* must occur later in filename's records than *item-name*.

   - If (*nn*) literal is specified, a format definition of a single literal comprising *nn* occurrences of the stated literal is generated; except that if the stated literal is a single space character, the generated literal is *nn*X. (If the stated literal is two or more spaces, the generated literal is nn occurrences of those spaces.)

10. If the ADABAS Interface is installed, the truncation limits of the ENTRY-POINT keyword of the SYSTEM, PROGRAM and MODULE data definition statements becomes:

    ```
    ENTRY-POINT
    ```

Example

See "Introduction" on page 3.

```
item-name [ { { ENTERED-AS     } [version]  [ { EDIT-NAME item-name-a [version] } ]
            { { HELD-AS        }              { EDIT-CODE 'En'                  }
            { { REPORTED-AS    }
            {   DEFAULTED-AS                ]
            {   version        }

 [ { COUNTS  [AS  item-name-c [version]]     } ] [IN group-name-a OCCURRENCES { g [,g]. . .           } ]
   { OCCURRENCES { i [,i]. . .          }    }                                { g { THROUGH } h       }
   {             { i { THROUGH }        } j  }                                {   { THRU    }         }
   {                 { THRU    }             }

group-name [ { ENTERED-AS   } [ { EDIT-NAME item-name-a [version] } ]
             { HELD-AS      }   { EDIT-CODE 'En'                  }
             { REPORTED-AS  }
             { DEFAULTED-AS }

           [ { COUNTS  [AS  item-name-c [version]                              } ]
             { OCCURRENCES { g [,g]. . .         }                             }
             {             { g { THROUGH }       } h                           }
             {                 { THRU    }                                     }
             { [IN group-name-a OCCURRENCES { g [,g]. . .          } ]         }
             {                              { g { THROUGH }      h }           }
             {                                  { THRU    }                    }

item-name { THROUGH   item-name-b }
          { THRU                  }

[(nn)] 'literal'

[KNOWN-AS local-name}
```

Figure 1. Format of details in PROCESSES ADABAS clause

ADABAS-related Aspects of the Item Data Definition Statement

Generation of ADABAS Attributes

The ITEM data definition statement (see the *ASG-DataManager User's Guide*) can include one of the ADABAS related form-description keywords VARIABLE, NULL-SUPPRESSED, or FIXED. If present, these keywords control the ADABAS attributes generated when the Source Language Generation facility is used to produce LOADER Definition statements; thus:

- VARIABLE results in the generation of a non-suppressed item
- NULL-SUPPRESSED results in the generation of the attribute NU (null)
- FIXED results in the generation of the attribute Fl (fixed).

If none of these keywords is present in the ITEM's form-description, the ITEM's length specification determines the attributes generated; thus:

- If the length is defined as p TO q, and p is greater than or equal to 1, the item is generated as non-suppressed
- If the length is defined as 0 TO q, the attribute NU (null) is generated
- If the length is defined as q, the attribute FI (fixed) is generated.

If the ENTERED-AS form of an item has an associated form-description of NULL-SUPPRESSED or variable length, then no length attribute is generated when LOADER Definition Cards are produced by the Source Language Generation facility.

Generation of BITS Items

When the Source Language Generation facility is used to generate LOADER Definition cards, any items that are defined to DataManager as BITS items are always generated as byte aligned binary fields, regardless of the value of the RNDBIT parameter in the DGADA macro. Thus, if any bit items are unaligned, the generated LOADER Definition cards may not be consistent with record layouts that are produced from the file.

4 ADABAS Source Language Generation from DataManager

Introduction

The DataManager Source Language Generation facility can be used to produce record layouts and ADABAS statements of the following types from encoded data definitions held in a DataManager data dictionary:

- LOADER Definition Cards. These provide the major part of the input required by the ADAWAN utility for database loading.
- COBOL, PL/I, or Assembler data description statements for:
  - Record buffers
  - Format buffers.

Generation of these statements is achieved by use of the PRODUCE command. The basic form of the command. which can generate record layouts or COBOL, PL/I or Assembler data descriptions for conventional file environments is described in the separate publication *ASG-Manager Products Source Language Generation*. Users should refer to that manual for a general description of source language generation and of the PRODUCE command. The variation of the command to produce ADABAS statements of any of the types stated above is described in "Specification of the PRODUCE Command for ADABAS Source Language Generation" on page 22.

An installation macro, DGADA, permits the output from these variations of the PRODUCE command to be tailored to conform to the particular installation’s standards. DGADA is described in the Appendix, "The Macro DGADA," on page 33. Certain parts of the output generated when producing ADABAS buffers can be further tailored by means of these DataManager installation macros:

- DGCOB, if the buffers’ data descriptions are produced in COBOL
- DGPLI, if the buffers’ data descriptions are produced in PL/I
- DGBAL, if the buffers’ data descriptions are produced in Assembler.

These macros are documented in the Source Language Generation manual.

In the specifications in this chapter, any ASG-defined conditions or values that can be tailored by the Controller are annotated "(unless tailored, see *xxxx*)", where *xxxx* is the relevant keyword of the appropriate macro, DGADA, DGCOB, DGPLI,or DGBAL.

Specification of the PRODUCE Command for ADABAS Source Language Generation

Format

```
PRODUCE

 ⎧ ADABAS LOADER-DEFINITIONS ⎫
 ⎩ RECORD-LAYOUTS            ⎭

     FROM filename [AS library-name
         [,filename [AS library-name]]. . .

[RECORD-LAYOUTS ⎧FOR⎫ ] ADABAS⎧ RECORD-BUFFERS ⎫
                ⎩AND⎭         ⎨ FORMAT-BUFFERS ⎬
                              ⎩ BUFFERS        ⎭

     [IN language]

     FROM ⎧  process-name [AS library-name]    ⎫
          ⎪ [,process-name [AS library-name]   ⎪
          ⎨  access-name [AS library-name]     ⎬
          ⎪ [,access-name [AS library-name]    ⎪
          ⎩  USED-IN process-name              ⎭

[control-options]   ⎧ ; ⎫
                    ⎩ . ⎭
```

where:

filename is the name of an encoded FILE ADABAS member

library-name is the name to be given to the generated library member in the output file. It must be not more than sixteen characters, of which the first character must be alphabetic or one of the characters #, £ (or local currency symbol with the internal code hexadecimal 5B), % or @.

language is any one of:

COBOL	PL/I
ASSEMBLER	PLI
BAL	PL/1
ALC	PL1

process-name is the name of an encoded SYSTEM, PROGRAM or MODULE member that has a PROCESSES ADABAS clause

access-name is an access-name defined in a PROCESSES ADABAS clause in the member named in the USED-IN clause

control-options is as defined in the Source Language Generation manual, except that:

output-form-1 in the GIVING clause is:

```
⎧ [s] NOTES              ⎫
⎪ [s] DESCRIPTIONS       ⎪
⎨ KNOWN-AS               ⎬
⎪ ACCESS-NAME-PREFIX     ⎪
⎩ UPDATES                ⎭
```

output-form-2 in the OMITTING clause is:

```
⎧ NOTES                  ⎫
⎪ DESCRIPTIONS           ⎪
⎨ ALIAS                  ⎬
⎪ KNOWN-AS               ⎪
⎪ ACCESS-NAME-PREFIX     ⎪
⎩ UPDATES                ⎭
```

```
GIVING ⎧ FD-ONLY      ⎫
       ⎨ RECORDS-ONLY ⎬
       ⎩ ALL-FILE     ⎭
```

are not relevant in this command.

Remarks

1. The first two elements of the command must be PRODUCE ADABAS, in that order. Other elements present in the command must be in the order shown under Format.

2. Up to a maximum of 16 *filenames* or *process-names* or *access-names* can be declared in the FROM clause. If two or more are declared, each except the first must be preceded by a comma; the comma can optionally be preceded and/or followed by spaces. Names are processed in the order in which they appear in the FROM clause.

3. Acceptance of the PRODUCE command is in respect of each *filename*, *process-name*, or *access-name* individually. If a member named in the FROM clause or in its subordinate USED-IN clause:

   - Is not encoded, or
   - Is not present in the data dictionary, or
   - Is of a member type that is not valid in the context, or
   - Is protected against access by the user (see remark 4),

   or if an *access-name* in the FROM clause is not defined in the member named in the subordinate USED-IN clause, then a message is output, no generation takes place in respect of that name, and processing continues with the next name or command.

4. Acceptance of the PRODUCE command is subject to access security levels (for each filename or process-name individually, as stated in remark 3). If a member named in the FROM clause or in its subordinate USED-IN clause has an access security level higher than the user’s (general or specific) security level, the command is rejected in respect of that member. If the command can be accepted in respect of a member, but a reference path from that member includes a protected member with an access level higher than the user’s security level, then:

   - If LOADER-DEFINITIONS are being generated the reference path is, if appropriate, followed to its end to determine the total storage space required, but the name of no member in that reference path beyond the last member to which the user has access is given; instead, a filler name is generated.
   - If record buffers and/or format buffers are being generated, and:
     - The protected member is directly referenced in a GIVING or USING clause in process-name (or in the member named in an AS clause of process-name; see "Specification of the PROCESSES Clause" on page 14), then it is omitted from the generated buffer (that is, no filler is generated in respect of the protected member)
     - The protected member is indirectly referenced from a GIVING or USING clause in `process-name` (or in the member named in an AS clause of `process-name`), then the reference path is, if appropriate, followed to its end to determine the total storage space required, but the name of no member in that reference path beyond the last member to which the user has access is given; instead, a filler name is generated.

5. If LOADER-DEFINITIONS is stated in the command, a set of LOADER Definition Cards is generated for each filename specified in the FROM clause. Fields may be generated from dictionary ITEM members. If an item is defined as being NUMERIC-CHARACTER (unpacked decimal) in form and over 27 bytes in length, then this is greater than the limit allowed for such a field (standard format U) by ADABAS. The DataManager Source Language Generation facility will generate such over-length numeric fields as alphanumeric (ADABAS standard format A) fields by default.

6. If RECORD-BUFFERS is stated in the command and is not preceded by RECORD-LAYOUTS FOR, data description statements are generated for the record buffers for each `process-name` or `access-name` specified in the FROM clause.

7. If FORMAT-BUFFERS is stated in the command and is not preceded by RECORD-LAYOUTS FOR, data description statements are generated for the format buffers for each `process-name` or `access-name` specified in the FROM clause.

8. If BUFFERS is stated in the command and is not preceded by RECORD-LAYOUTS FOR, data description statements are generated both for the record buffers and for the format buffers for each `process-name` or `access-name` specified in the FROM clause.

9. If RECORD-BUFFERS, FORMAT-BUFFERS or BUFFERS is stated in the command, the language in which the data description statements are generated is determined by process-name’s LANGUAGE clause unless overruled by an IN clause in the command. Languages recognised are as stated for language under Format above. If (with RECORD-BUFFERS FORMAT-BUFFERS, or BUFFERS stated in the command):

   - Any other language is specified in the IN clause of the command, or
   - There is no IN clause in the command and `process-name` has no LANGUAGE clause, or has a LANGUAGE clause that specifies some other language.

   then an error message is output, no generation takes place and DataManager proceeds to the next command.

10. AS clauses are relevant only if ADABAS LOADER Definition Cards or buffer data description statements are being produced and written to an output data set. (Statements can be generated for listing on a printer or terminal only, without being written to an output data set; see the `control-options` specifications.)

11. Each AS clause present in the command relates only to the `filename`, `process-name`, or `access-name` that immediately precedes it. It declares a name under which the generated LOADER Definition or source language data description is to be catalogued in the output source library data set.

12. For each `filename`, `process-name`, or `access-name` for which no AS clause is specified, `library-name` is defaulted to `filename`, `process-name`, or `access-name` respectively if the name conforms to the length restriction on `library-name`. The length restriction on `library-name` is a maximum of eight characters (unless tailored, see MEMLEN). If the name is longer than the permitted maximum length for `library-name`, no generation takes place in respect of that name, a message is output, and processing continues with the next name or command.

13. `Library-names`, whether declared or defaulted, are not subject to any name editing, ALIAS or WITH-ALIAS clauses (see `control-options`) that may be present in the command.

14. If RECORD-LAYOUTS FROM filename is stated in the command, record layouts are generated from the encoded FILE ADABAS members specified in the FROM clause. If RECORD-LAYOUTS FOR is stated followed by RECORD-BUFFERS, FORMAT-BUFFERS, or BUFFERS, then record layouts for the appropriate buffers are generated from each `process-name` or `access-name` specified in the FROM clause. If RECORD-LAYOUTS AND is stated followed by RECORD-BUFFERS, FORMAT-BUFFERS, or BUFFERS, then both record layouts and data description statements are generated for the appropriate buffers.

15. The control-options clause GIVING ACCESS-NAME-PREFIX overrides the ACCNAM=NO keyword usage of the DGADA macro. (The supplied value of the ACCNAM keyword is YES.) Generated names for fields in buffers are prefixed with `access-name`.

16. The `control-options` clause OMITTING ACCESS-NAME-PREFIX overrides the ACCNAM=YES keyword usage of the DGADA macro. Generated names for fields in buffers are not given a prefix of `access-name`.

17. The control-options clause GIVING UPDATES overrides the UPDATES=NO keyword usage of the DGADA macro. (The supplied value of the UPDATES keyword is YES.) The file member’s source record will be updated as explained in remark 21.

18. The control-options clause OMITTING UPDATES overrides the UPDATES=YES value of the DGADA macro. No file member’s source record will be updated by the PRODUCE command.

19. If SEQUENTIAL is stated in the *`output-control-options`* and *`control-card`* is not stated (see the publication *ASG-Manager Products Source Language Generation*), then no control card is output (unless tailored, see CONCARD).

20. Other *`control-options`* keywords and clauses are as stated in the publication *ASG-Manager Products Source Language Generation*.

21. The FILE member specified in:

    - The FROM clause, if producing LOADER Definition Cards, or
    - The ACCESS clause of *`process-name`* (or in the member named in an AS clause of *`process-name`*), if producing buffer data description statements

    is examined to ensure that an entry appears in its FIELD-NAMES clause (see "The File Data Definition Statement for ADABAS Files" on page 10) for every group and item that is directly or indirectly referenced from its CONTAINS clause. If any such entry is missing DataManager generates an adabas-name for the group or item, as described in remark 23. If the PRODUCE command is issued in a non-frozen status and the user has sufficient authority to alter and to re-encode the FILE member, then, subject to remarks 17, 18 and 22, (or unless tailored, see UPDATES), for each DataManager-generated *`adabas-name`*, DataManager inserts a separate clause at the end of the FILE member’s source record, in this format:

    ```
    FIELD-NAMES adabas-name IS ⎧group-name ⎫
                               ⎩item-name  ⎭

    DMR-NOTE 'GENERATED BY DataManager AT hh.mm ON dd mm yy'
    ```

    The member is then re-encoded. This ensures that the same *`adabas-names`* are used from one PRODUCE command to the next, and thus that the same *`adabas-names`* are used for LOADER Definition cards as for buffer data description statements.

22. If the user has insufficient authority to re-encode the FILE member, then no member’s source record is updated by the PRODUCE command. Thus, if any member directly referenced by any FILE member that is specified in the command as stated in remark 21 cannot be accessed by the user, then no insertion of FIELD-NAMES clauses is performed by the PRODUCE command.

23. Any *`adabas-names`* generated as stated in remark 21 are generated thus:

   - The *`member-name`*, *`alias`*, or *`local-name`* is selected, as determined by *`control-options`* clauses of the PRODUCE command and the keyword values of the macro DGADA
   - If producing LOADER Definition cards, any editing specified in editing clauses of the *`control-options`* is applied
   - The resulting name is submitted to the final editing described in remark 24.

24. After all editing specified in editing clauses of the *`control-options`* has been completed, DataManager performs a final automatic editing of all *`adabas-names`* generated for use in LOADER Definition cards or buffer data definition statements, thus:

   - If the name is longer than two characters, it is reduced to two characters by removing middle characters
   - If the first character of the resulting name is non-alphabetic, or if the name is otherwise an illegal ADABAS name, or if it duplicates a previous *`adabas-name`* output for this FILE member or an *`adabas-name`* specified in the FILE member, a filler name is substituted.

25. If buffer data description statements are produced, then after all editing specified in the DGADA macro and all editing specified in editing clauses of the *`control-options`* has been completed, DataManager performs a final automatic editing of generated names (other than *`adabas-names`*, see remark 24) to ensure conformity with the rules of the relevant source language, by:

   - Removing any characters that are illegal in the particular source language
   - Shortening any names that are longer than the maximum permitted in the particular source language by removing middle characters.

Example

For examples of the PRODUCE command and the resulting output, see the *ASG-DataManager Example Book*.

5 DataManager/ADABAS Correspondence Tables

The tables in this chapter indicate the correspondence between ADABAS LOADER Definition Cards and DataManager data definitions, and between ADABAS Format Buffers and DataManager data definitions.

ADABAS/DataManager Correspondence: LOADER Definition Cards

ADABAS	DataManager
`level number`	`Generated by PRODUCE command`
`fld-name`	`item-name or group-name from CONTAINS or FIELD-NAMES clauses of a FILE`
`std-length`	`ADABAS member`
	`ITEM length*`
`std-length not specified or zero`	`ITEM-ENTERED-AS* variable length`
	`ITEM-ENTERED-AS NULL-SUPPRESSED`
`std-format A`	`ITEM CHARACTER*`
`std-format B`	`ITEM BINARY*`
`std-format G`	`ITEM FLOATING-POINT*`
`std-format P`	`ITEM PACKED-DECIMAL*`
`std-format U`	`ITEM NUMERIC-CHARACTER*`
`def-option`	`FILE ADABAS`
`DE`	`DESCRIPTORS item-name`
`FI`	`CONTAINS item-name  (not variable length)`
	`(Not NUMERIC-CHARACTER, unless tailored. See NUMCHAR.)`
`NU`	`CONTAINS item-name (variable length, minimum 0)`
	`(NUMERIC-CHARACTER unless tailored. See NUMCHAR.)`
`default length option`	`CONTAINS item-name (veriable length, minimum 0)`

ADABAS/DataManager Correspondence: LOADER Definition Cards

ADABAS	DataManager

`MU (N)`
`MU`
	`CONTAINS (` $\left\{\begin{matrix}\texttt{integer}\\\texttt{item-name}\end{matrix}\right\}$ `) item-name`

`PE (n)`
`PE`
	`CONTAINS (` $\left\{\begin{matrix}\texttt{integer}\\\texttt{item-name}\end{matrix}\right\}$ `) group-name`

`optional user comments normally COBOL or English language field name`	`Member name of the GROUP or ITEM directly or indirectly contained by the FILE ADABAS member.`

* These attributes belong to LOADER INPUT.

Note:

LOADER Definition Cards generated from DataManager data definitions may include fields generated from dictionary ITEM members. If an item is defined as being NUMERIC-CHARACTER, with length over 27 bytes, then this is greater than the limit allowed for such a field by ADABAS. DataManager will generate such fields as alphanumeric by default.

In this table, DataManager elements are from the details (see Figure 1 on page 19) specified for a PROCESSES clause of a SYSTEM, PROGRAM, or MODULE member, unless otherwise stated.

ADABAS/DataManager Correspondence: Format Buffers

ADABAS	DataManager
`fld-name`	`adabas-name from FIELD-NAMES clause of a FILE ADABAS member`
`fld-name i`	$\left\{\begin{matrix}\texttt{item}\\\texttt{group}\end{matrix}\right\}$ `OCCURRENCE i`

`fld-name i,`
`fld-name j`
	$\left\{\begin{matrix}\texttt{item}\\\texttt{group}\end{matrix}\right\}$ `OCCURRENCES i,j`

`fld-name i-j`	$\left\{\begin{matrix}\texttt{item}\\\texttt{group}\end{matrix}\right\}$ `OCCURRENCES i` $\left\{\begin{matrix}\texttt{THROUGH}\\\texttt{THRU}\end{matrix}\right\}$ `j`

`fld-name g(i-j),`
`fld-name h(i-j)`
	`item OCCURRENCES i` $\left\{\begin{matrix}\texttt{THROUGH}\\\texttt{THRU}\end{matrix}\right\}$ `j`
	`IN group-a OCCURRENCES g,h`

`fld-name g-h (i-j)`	`item OCCURRENCES i` $\left\{\begin{matrix}\texttt{THROUGH}\\\texttt{THRU}\end{matrix}\right\}$ `j`
	`IN group-a OCCURRENCES g` $\left\{\begin{matrix}\texttt{THROUGH}\\\texttt{THRU}\end{matrix}\right\}$ `h`

ADABAS/DataManager Correspondence: Format Buffers

ADABAS	DataManager

`fld-name g(i),`
`fld-name g(j),`
`fld-name h(i),`
`fld-name h(j)`
	`item OCCURRENCES i,j`
	`    IN group-a OCCURRENCES g,h`

`fld-name -`
`fld-name`
	`item` $\left\{\begin{matrix}\texttt{THROUGH}\\\texttt{THRU}\end{matrix}\right\}$ `item-b`

`fld-nameC`	$\left\{\begin{matrix}\texttt{item}\\\texttt{group}\end{matrix}\right\}$ `COUNTS [AS item-name-c]`
`nX`	`(nn)' '`
`'literal'`	`(nn)'literal'`

`length`
`format`
	`from form-description of the form/version of`
	$\left\{\begin{matrix}\texttt{item}\\\texttt{group}\end{matrix}\right\}$ `indicated in details if different`
	`from the form/version in the FILE member.`

`E1 to E15`	`EDIT-CODE 'En' or EDIT-NAME item-name-a (via`
	`CONTENTS clause).`

Appendix
The Macro DGADA

The macro DGADA enables the generation of ADABAS source statements by the PRODUCE command of DataManager to be tailored to conform to a particular installation's standards.

The macro DGADA is supplied as a source module on the installation magnetic tape. The table below lists the keywords of this macro, for which values can be specified when DataManager is installed. If the supplied default values of all of these keywords are acceptable, no further action need be taken in respect of this macro. If any values are to be changed, the procedure described in the publications *ASG-Manager Products Installation in OS Environments* or *ASG-Manager Products Installation in DOS Environments* must be followed. The macro assembles as the DataManager module DFU16.

The Macro DGADA: Keywords Specifiable on Installation

Keyword	Specifies	Default Value	Alternate Value
ACCNAM	Whether *`access-name`* to be prefixed to names of buffers and their constituent fields.	YES	NO
ACHAR	The hexadecimal values of any additional characters that are to be accepted for output in names produced by the Source Language Generation facility, to enable characters not in the standard source language character set to be output (see Note 1).	No default	Any valid hexadecimal value, or a sublist of such values.
ACSMETH	The type of file to be generated by a PRODUCE command.	BPAM	QSAM
ALIAS	Whether ADABAS specific aliases are to be generated instead of member names.	NO	YES (see Note 2)
ALLFILL	Whether all names of fields in buffers to be filler names.	NO	YES
ANFILL	Whether the filler name from FSTART is to be increased alphanumerically in the junior position (i.e., X0, X9, XA, XZ, Y0) instead of numerically (i.e., X0, X9, Y0, Y9).	YES	NO (i.e., numeric increments)

The Macro DGADA: Keywords Specifiable on Installation

Keyword	Specifies	Default Value	Alternate Value
AUTOCHK	Check for and convert fillers.	YES	NO
CNTLIT	Count literal to be added to buffer names.	'COUNT'	Any 1- to 32-character name within delimiters
CNTPRE	Prefix names of fields in buffers with value of CNTLIT (when appropriate).	NO	YES
CNTSUF	Suffix names of fields in buffers with value of CNTLIT (where appropriate).	YES	NO
COLNOTE	Column number where DataManager comment of "real" member name commences.	46	An integer greater than 46
CONCARD	Whether a control card is to be produced.	NO	YES (see Note 3)
DDNAME	Default library name.	'GENLIB'	A delimited string of 1 to 8 characters
DESC	Reserved for future use.	NO	None
FBUFLIT	Literal to be added to names of fields in format buffer.	'FB'	Any 1- to 32-character name within delimiters
FBUFPRE	Prefix names of fields in format buffer with value of FBUFLIT.	NO	YES
FBUFSUF	Suffix names of fields in format buffer with value of FBUFLIT.	YES	NO
FORBUF	Name for format buffer.	'FORMAT-BUFFER'	Any 1- to 32-character name within delimiters
FSTART	The two-digit ADABAS name to be used as the start name when generating filler names.	'X0'	Any valid delimited ADABAS name
HYPHEN	Whether hyphens are to be added to prefixes and suffixes.	YES	NO
KNOWNAS	Whether *`local-names`* from KNOWN-AS clauses are to be generated instead of member names.	NO	YES
LIBCC	The format of the control card output as the first card of a QSAM file (unless overridden by *`control card`* in an ONTO clause).	See the Source Language Generation manual.	A delimited character string of 1- to 72-characters including question mark (?)
MAXLEV	Maximum ADABAS level number.	7	None (see Note 4)

The Macro DGADA: Keywords Specifiable on Installation

Keyword	Specifies	Default Value	Alternate Value
MAXLIT	Maximum literal length for Format Buffer literals.	40	20 to 255
MAXLOAD	The maximum number of definition cards to be produced before an error message is issued.	500	Any valid number
MAXMU	Maximum number of multiple values in an MU field.	191	None (see Note 4)
MAXPE	Maximum number of occurrences in a periodic group.	99	None (see Note 4)
MEMLEN	Maximum length of *library-name*.	8	Up to 16
NONAME	Whether *member-name*, *alias*, or *local-name* element to be omitted from names of fields in buffers.	NO	YES
NOTE	Reserved for future use.	NO	None
NUMCHAR	Whether NUMERIC-CHARACTER form-descriptions in ITEM members are to generate NU or FI options on LOADER Definition Cards.	NU	FI
RBUFLIT	Literal to be added to names of fields in record buffer.	'RB'	Any 1- to 32-character name within delimiters
RBUFPRE	Prefix names of fields in record buffer with value of RBUFLIT.	NO	YES
RBUFSUF	Suffix names of fields in record buffer with value of RBUFLIT.	YES	NO
RECBUF	Name for record buffer.	'RECORD-BUFFER'	Any 1- to 32-character name within delimiters
RNDBIN	Whether binary items are to be rounded.	NO	YES
RNDBIT	Whether bit string fields should be generated with byte alignment (see Note 5).	YES	NO
UNIQ	Whether a number is to be inserted immediately following the values of FBUFLIT and RBUFLIT to make field names unique (if necessary).	YES	NO (number will be added on end of name)
UPDATES	Whether a file member's source record is to be updated by the PRODUCE command.	YES	NO

Notes

1. The standard Source Language Generation facility output character set for the ADABAS Interface is that defined in the ADABAS data definition language specification. This character set can be extended to allow non-standard characters to be output in names, by entering the hexadecimal value of each required character as a value to ACHAR. The user should ensure that any extra characters that are added to the output character set in this way are used only in ways that are permitted by the software with which DataManager is used.

2. If ALIAS=YES and KNOWNAS=YES both apply, then when a data name is generated for a member that has an ALIAS clause and is subject to a containing member's KNOWN-AS clause, the KNOWN-AS *local-name* takes precedence.

3. When the value CONCARD=NO is used, to suppress the generation of a control card, the production of BKEND cards is also suppressed.

4. The values of MAXLEV, MAXMU, and MAXPE are ADABAS constants for which no alternative values are available. Provision has been made in the macro for declaring alternative values in case ADABAS should permit alternative values in the future.

5. The effect of the RNDBIT parameter is overridden by any alignment specification stated in the data definition of any group, record, or file that contains the bit string item.

Index